Web Scraping for Data Science with Python

Seppe vanden Broucke and Bart Baesens

– Free Extract –

Get the full book on Amazon

9.15 Scraping and Visualizing a Board Members Graph

In this example, our goal is to construct a social graph of S&P 500 companies and their interconnectedness through their board members. We'll start from the S&P 500 page at Reuters available at *https://www.reuters.com/finance/markets/index/.SPX* to obtain a list of stock symbols:

```
from bs4 import BeautifulSoup
import requests
import re

session = requests.Session()

sp500 = 'https://www.reuters.com/finance/markets/index/.SPX'

page = 1
regex = re.compile(r'\/finance\/stocks\/overview\/.*')
symbols = []

while True:
    print('Scraping page:', page)
    params = params={'sortBy': '', 'sortDir' :'', 'pn': page}
    html = session.get(sp500, params=params).text
    soup = BeautifulSoup(html, "html.parser")
    pagenav = soup.find(class_='pageNavigation')
    if not pagenav:
        break
    companies = pagenav.find_next('table', class_='dataTable')
    for link in companies.find_all('a', href=regex):
        symbols.append(link.get('href').split('/')[-1])
    page += 1

print(symbols)
```

Once we have obtained a list of symbols, we can scrape the board member pages for each of them (e.g. *https://www.reuters.com/finance/stocks/company-officers/MMM.N*), fetch the table of board members, and store it as a pandas data frame, which we'll save using pandas' `to_pickle` method . Don't forget to install pandas first if you haven't already:

```
pip install -U pandas
```

Add this to the bottom of your script:

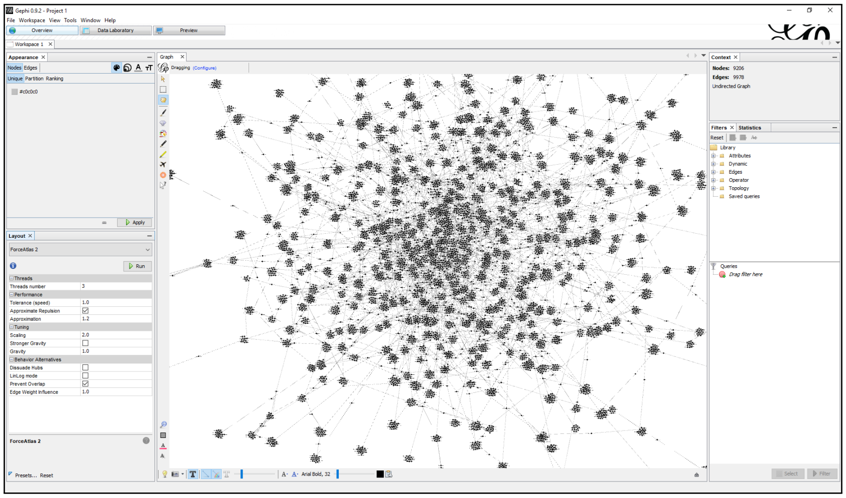
```
import pandas as pd

officers = 'https://www.reuters.com/finance/stocks/company-officers/{symbol}'

dfs = []

for symbol in symbols:
    print('Scraping symbol:', symbol)
    html = session.get(officers.format(symbol=symbol)).text
    soup = BeautifulSoup(html, "html.parser")
    officer_table = soup.find('table', {"class" : "dataTable"})
    df = pd.read_html(str(officer_table), header=0)[0]
    df.insert(0, 'symbol', symbol)
    dfs.append(df)

# Store the results
df = pd.concat(dfs)
df.to_pickle('sp500.pkl')
```

This sort of information can lead to a lot of interesting use cases, especially—again—in the realm of graph and social network analytics. We're going to use NetworkX once more, but simply to parse through our collected information and export a graph in a format which can be read with Gephi, a popular graph visualization tool, which can be downloaded from *https://gephi.org/users/download/*:

```
import pandas as pd
import networkx as nx
from networkx.readwrite.gexf import write_gexf

df = pd.read_pickle('sp500.pkl')

G = nx.Graph()

for row in df.itertuples():
    G.add_node(row.symbol, type='company')
    G.add_node(row.Name,type='officer')
    G.add_edge(row.symbol, row.Name)

write_gexf(G, 'graph.gexf')
```

Open the graph file in Gephi, and apply the "ForceAtlas 2" layout technique for a few iterations. We can also show labels as well, resulting in a figure like in Figure 9.15.

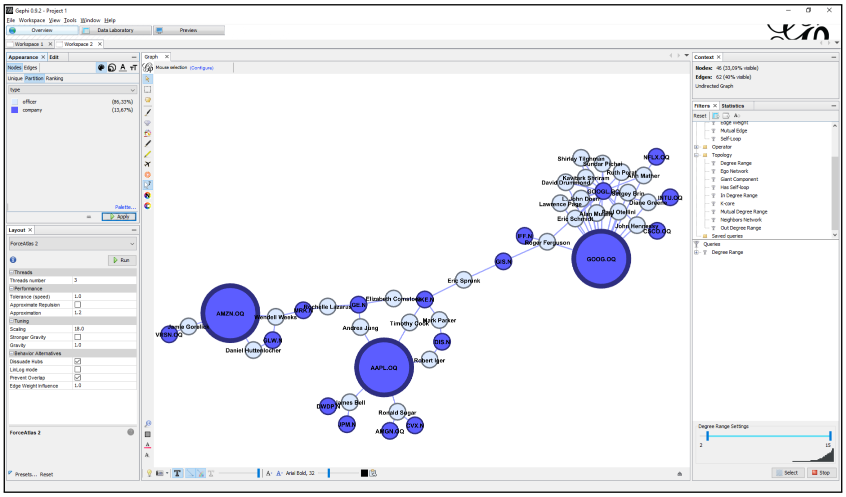
Take some time to explore Gephi's visualization and filtering options if you like. All attributes that you have set in NetworkX ("type", in our case) will be available in Gephi as well. Figure 9.16 shows the filtered graph for Google, Amazon and Apple with their board members which are acting as connectors to other firms.



Nodes: 9206
Edges: 9978
Undirected Graph

ForceAtlas 2
Threads number 3
Tolerance (speed) 1.0
Approximate Repulsion ✓
Approximation 1.2
Scaling 2.0
Stronger Gravity
Gravity 1.0
Dissuade Hubs
LinLog mode
Prevent Overlap ✓
Edge Weight Influence 1.0

Figure 9.15: Visualizing our scraped graph using Gephi.



Nodes: 46 (33,09% visible)
Edges: 62 (40% visible)
Undirected Graph

type
officer (86,33%)
company (13,67%)

ForceAtlas 2
Threads number 3
Tolerance (speed) 1.0
Approximate Repulsion
Approximation 1.2
Scaling 18.0
Stronger Gravity
Gravity 1.0
Dissuade Hubs ✓
LinLog mode
Prevent Overlap ✓
Edge Weight Influence 1.0

Figure 9.16: Showing connected board members for Google, Amazon, and Apple.

Web Scraping for Data Science with Python

Seppe vanden Broucke and Bart Baesens